DOCUMENT RESUME

ED 178 322 SE 029 205

AUTHOR White, James W.
TITLE An Off-Line Simulation System for Development of Real-Time FORTRAN Programs.
INSTITUTION Arizona Univ., Tucson. Dept. of Chemical Engineering.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 74
GRANT NSF-SED-74-21344
NOTE 43p.

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Programs; Computers; *Program Development; Programing; Programs; *Simulation; *Technical Reports
IDENTIFIERS *FORTRAN; *Minicomputers

ABSTRACT

Implementation of an ISA FORTRAN standard for executive functions and process input-output within a simulation system called MINIFOR provides a useful real-time program development tool for small single function, dedicated minicomputers having a FORTRAN compiler but limited program development aids. A FORTRAN-based pre-compiler is used off-line to eliminate most compilation errors, while the run-time system is used off-line to eliminate most execution errors. The description of the structure and features of the MINIFOR system illustrates its capabilities and limitations. Attention is directed toward the run-time system and an example of its use is included. (Author/MK)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Mary L. Charles
NSF

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED178322

SE 029 205

AN OFF-LINE SIMULATION SYSTEM FOR DEVELOPMENT OF REAL-TIME FØRTRAN PROGRAMS

James Wm. White

Department of Chemical Engineering

College of Mines
University of Arizona
Tucson, Arizona 85721

SUMMARY

There are two common approaches to developing or updating a dedicated minicomputer FØRTRAN program. Usually, the control program is coded and compilation debugged on a large-scale scientific computing system or the minicomputer itself if the machine is available. Final debugging and testing must wait for installation and checkout of the entire minicomputer system. The need to obtain, install and check out the hardware before program development can be completed as well as interference with normal process operation during final testing or program revision are two clear disadvantages of these approaches. The MINIFØR program development system described here is really most useful for applications where the minicomputer has only a minimum input/output configuration of paper tape and teletype and is especially useful for teaching basics of real-time computing to undergraduate engineering students. To promote portability all MINIFØR modules are coded in ANSI FØRTRAN IV. The real-time process structure of MINIFØR is realized by host computer simulation of a real-time clock, a process input/output interface and an approximate mathematical model of the process in state variable form. In order to eliminate compilation errors specific to a particular minicomputer, MINIFØR includes a general purpose minicomputer pre-compiler generation system. After a successful pre-compilation (scanning and

parsing phases only) the user then "executes" the minicomputer FØRTRAN application program for the actual process installation against this simulation of the process and minicomputer system. MINIFØR uses the process input/output interface standards established by the Fifth Purdue Workshop on Standardization of Industrial Computer Languages and issued as Standard S61.1 by ISA[15]. Hence, assembly language handlers for these standards must be available in the minicomputer library. Exceptions and extensions to ANSI FØRTRAN IV are specified at pre-compiler set up time. Pre-compiler generation is only repeated if the minicomputer FØRTRAN dialect changes.

INTRODUCTION

The phenomenal growth of the minicomputer industry in the last decade has resulted from the continuing decrease in the cost of components. Progress in the semiconductor and memory technology fields has been the primary contributor, reducing minicomputer main frame cost significantly. A leading minicomputer manufacturer recently advertised[13] that in 1965 one of its models cost $18,000, and today a more reliable version of the same model costs only $4,000. Each of 77 minicomputer models surveyed by Murphy[5] was available in the $2,000 to $23,000 price range. This price included a standard processing unit, basic memory, and occasionally included as standard equipment a teletype,

low speed paper tape reader-punch, and real-time clock. This low cost has now made installation of a small computer very attractive, whereas previously computer-based alternatives to conventional data acquisition and control systems were uneconomical. Minicomputer annual shipments are forecast by Kaenel[3] to grow from 6,000 units in 1969 to 40,000 units by 1975. Approximately half of these units will require real-time capabilities for data-acquisition and process control applications. Later surveys[14] have largely substantiated these predictions.

Motivation

Each minicomputer is programmable in its own assembly language. Thus, programs developed on one machine are in general not transferable to another machine without extensive modification. While the existence of emulator and translator programs does ease assembly language programming, Williams[11] has stated that their use is restricted to the assembler mode of programming where the minicomputer to be programmed is usually emulated on another computer of the same manufacturer. Assembly language is more efficient from the standpoint of memory requirements and execution time, but these advantages are decreasing as minicomputer main frame costs and cycle times continue to drop. This is especially true in areas where memory requirements or

timing considerations are not critical. In the early stages of system development, companies such as General Electric, Honeywell, and Westinghouse, as reported by Jarvis[2], Mensh and Diehl[4], and Roberts[7], respectively, found standard FØRTRAN IV inadequate for real-time applications. However, because FØRTRAN had such influence in the engineering community and had become, in effect, a defacto standard, these companies as well as others included extensions to FØRTRAN IV to facilitate real-time programming in a compiler level language.

Standardization of some of these extensions has been proposed by the FØRTRAN Committee of the Workshop on Standardization of Industrial Computer Languages[16,17,18,19,20]. The Instrument Society of America (ISA) recently accepted some of these extensions[15]. Several mini-manufacturers now offer a full line of ISA standard handlers and one engineering company has developed extensive routines to test the conformity of these handlers to the published ISA standards.

Efficiency of real-time FØRTRAN program development suffers on a small computer for several reasons. Efficient compiler operation cannot be realized in less than 8K of core memory though many 4K versions do exist. Even at the 8K level, small computer FØRTRAN compilers are extremely

simple and do not produce extensive diagnostics as do compilers on large computer systems. The classic debugging aid is the run-time memory dump which prints the core contents at a stage of computation where execution is unexpectedly halted. However, the memory dump is not practical for small machines with very limited input/output devices. Spencer, Shepardson, and McGowan[9] report that most minicomputers marketed have available basic program development software consisting of an assembler, text editor, loader, and an on-line debugging aid. Of the minicomputer models surveyed by Murphy[5], 54 had FØRTRAN compilers. Program development on a minimum machine configuration of paper tape and teletype is awkward and time consuming even with the software aids previously mentioned. If control of a process requires the dedication of a small computer, then on-line program development, debugging, and updating may not be practical. Thus, a need exists for an application system which allows the user to accomplish most program development and debugging off-line on a large general purpose machine where extensive compiler diagnostics and debugging aids can be utilized.

An Alternative

This paper describes an application system that can be used off-line for program development, debugging, and updating of real-time FØRTRAN IV programs. The system, here-

after called MINIFØR, is an aid in dedicated minicomputer real-time data acquisition and process control applications which have a minimum input/output configuration of paper tape and teletype. MINIFØR has proven especially useful in teaching basic concepts of real-time computing to undergraduate engineering students where only limited configuration or limited access systems are available. The system includes a card-to-tape conversion program which produces FØRTRAN IV source programs in ASCII (American Standard Code for Information Interchange) code on paper tape. The source program can be transported to the minimum configuration small computer for utilization in an actual real-time application. Since MINIFØR itself and the card-to-tape conversion program are coded in ANSI (American National Standards Institute) FØRTRAN IV, the MINIFØR system is essentially large machine independent. Jobs can be processed on most large general purpose computer systems in either a batch or time-sharing mode. The principal features of MINIFØR are summarized in Table I.

TABLE I. MINIFØR Features

- ALL CODING IN ANSI FØRTRAN IV

  to decrease the problems inherent in program transfer and use.

- GENERAL PURPOSE PRE-COMPILER GENERATION

  to eliminate minicomputer compilation bugs not trapped by the more powerful host computer compilers

- RUN TIME EXECUTIVE

  to eliminate execution bugs prior to use in the actual process, laboratory, or instruction facility.

- HANDLES ALGEBRAIC OR DYNAMIC MODELS

  so that both data acquisition (e.g. chromatograph analysis) and transient problems (e.g., process control) can be handled with equal ease.

- TWO FIXED (RK2, RK4) AND ONE VARIABLE (RKM) INTEGRATION ROUTINES

  permit user selection of the integration algorithm for dynamic models

- PURE TIME DELAY FUNCTIONS PROVIDED

  to accommodate the large transport lags associated with some processes

- PROCESS I/Ø WRITTEN TO ISA STANDARD S61.1

  to promote program interchange between the different minicomputers now supporting these calls.

- FLEXIBLE INTERFACE HARDWARE SPECIFICATIONS

  to allow definition of a wide range of process/ computer interfaces

- TABULAR SPECIFICATION OF NON-LINEAR TRANSDUCERS AND FINAL CONTROL ELEMENTS

  to represent accurately a wide range of process hardware

- EASY-TO-USE STANDARD CODING FORMS

  to help the user specify his particular system features

- EXTENSIVE SET-UP AND RUN TIME DIAGNOSTICS

  for aid in debugging a user application

- DETAILED SYSTEM SUMMARY REPORTS AND PLOTS

  to keep track of all information passed between the process model and the user application program.

- STAND-ALONE CARD-TO-TAPE CONVERSION PROGRAM

  to generate even or odd parity ASCII paper tapes on host systems equipped with a paper tape punch.

Further work is currently underway to increase the generality and utility of the MINIFØR system. These proposed features are summarized in Table II.

TABLE II. MINIFØR Extensions Currently in Progress

- DEVELOPMENT of a CONVERSATIONAL USER SYSTEM SPECIFICATION MODULE

  to facilitate system set-up in a time-sharing environment.

- ADDITION OF AN AVAILABLE[8] FØRTRAN-TO-FØRTRAN OPTIMISING COMPILER

  to revise user developed code for better minicomputer core economy and lower execution time.

- IMPLEMENTATION OF PROPOSED[19] ISA S61.2 HANDLERS FOR RANDOM ACCESS FILE I/Ø AND PROPOSED[20] ISA S61.3 HANDLERS FOR MULTITASKING

  to provide a full set of the real-time functions so commonly used in modern minicomputer application programs.

MINIFØR can be utilized by industry to develop and update real-time programs for data acquisition and process control applications. The student can use MINIFØR to investigate or specify a process control strategy for a designated mathematical process model supplied by the instructor. In laboratories where desired unit operations equipment is not available, MINIFØR can be used in a data acquisition or control application to emulate the process operation needed. In well equipped laboratories where a minicomputer and interfacing hardware are available, the student can utilize MINIFØR to develop real-time programs for laboratory

exercises without the need to test the program under development against the actual process. This technique is used in the Department of Chemical Engineering at the University of Arizona to support a course entitled "Real-Time Computing in the Process Industries".

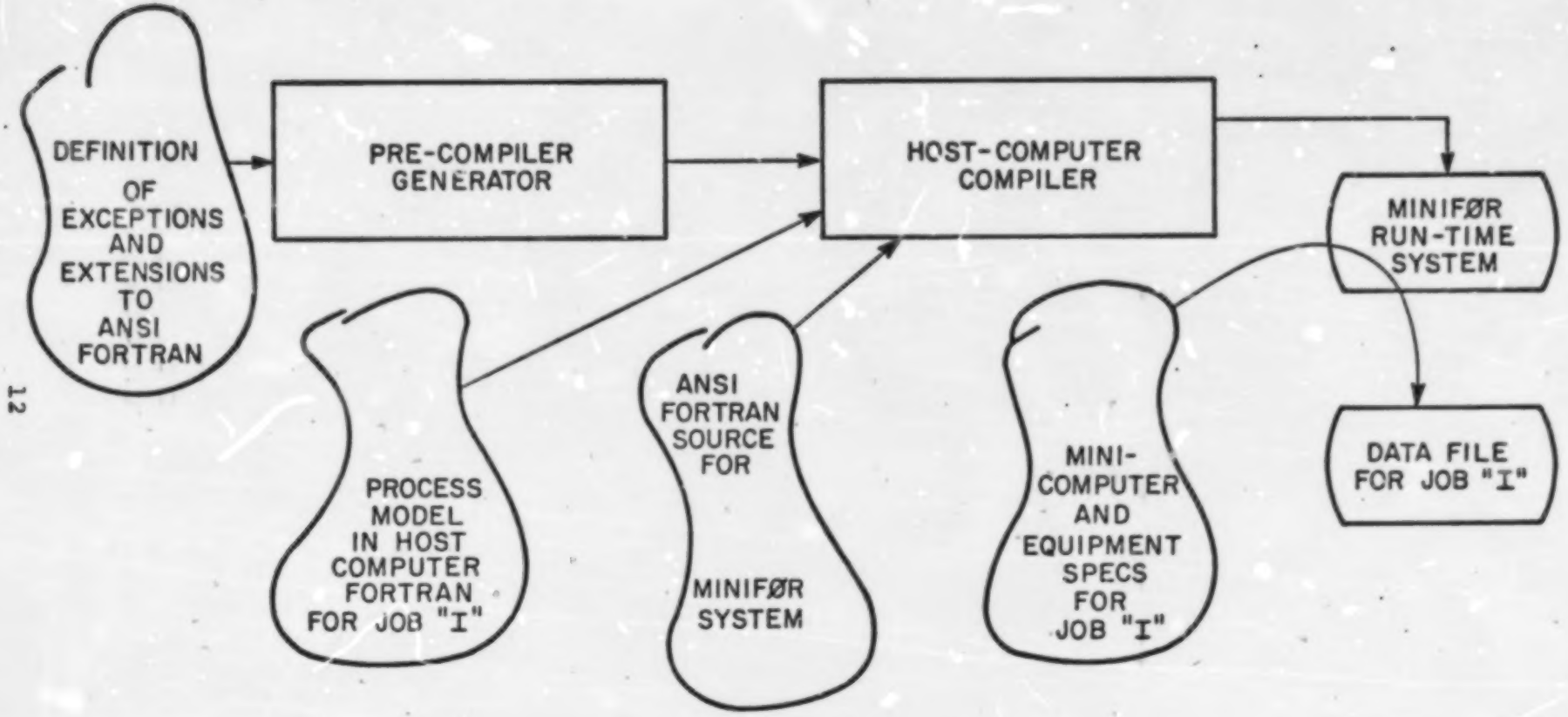
Processing Performed

MINIFØR processes an application program and any subroutines required by the application program. In order to compilation and execution debug and develop a real-time application program off-line, a minicomputer precompiler as well as a process simulation executive must be provided. A true cross compiler would be preferred of course but few are available. The run-time simulator must operate on a real-time basis and include communication to and/or from the process through interfacing hardware. This is accomplished by MINIFØR through simulation of the process interface. The User provides a model of the process, an application program to operate on the process, and input data to describe the simulation and hardware specifications. The process description may be wholly algebraic, or it may include dynamic process simulation capability using first-order ordinary differential equations in state variable form. To solve a dynamic model, MINIFØR calls on one of several system integration routines selected by the User to update the

process simulation as required by the pseudo real-time input/output subroutines. MINIFØR allows the User to program disturbances in the form of step changes entering the process at specified times. Also, dead time can be inserted for selected variables. To increase the relevance of the simulation, transducer and final control element characteristics are completely defined as part of the input data.

An overview of the MINIFØR system set-up sequence for a time sharing installation appears in Figure 1. The pre-compiler is generated only once for a particular minicomputer unless the manufacturer updates or alters the FØRTRAN dialect. Exceptions and/or extensions to ANSI FØRTRAN IV are input using a special MINIFØR precompiler set-up module. Scanner options; semantic action routines; or additions, deletions or modifications of a system file which defines ANSI FØRTRAN IV syntax in BNF (Backus Naur Form) can be specified. The set-up sub-system writes a dialect minicomputer BNF file which is then accessed by an SLR(1) parser generator. The output of the parser generator is assembled and stored for use by a pre-compiler driver module. During pre-compilation, the User's FØRTRAN dialect source code is scanned and parsed. Detailed diagnostics are output but Minicomputer object code is not generated. All pre-compiler modules are written in ANSI FØRTRAN IV. Details of the FØRTRAN-based scanner, parser generator and driver are

MINIFØR SYSTEM SET-UP


DEFINITION OF EXCEPTIONS AND EXTENSIONS TO ANSI FORTRAN
PRE-COMPILER GENERATOR
HOST-COMPUTER COMPILER
MINIFØR RUN-TIME SYSTEM
PROCESS MODEL IN HOST COMPUTER FORTRAN FOR JOB "I"
ANSI FORTRAN SOURCE FOR MINIFØR SYSTEM
MINI-COMPUTER AND EQUIPMENT SPECS FOR JOB "I"
DATA FILE FOR JOB "I"

14 15

FIGURE 1

provided elsewhere[10]. The MINIFØR run time system need only be compiled once unless local mods are desired. The model must be compiled and the computer/process specs supplied for each job supported. Actual use of Version 3.1 of the MINIFØR system after set-up is illustrated in Figure 2.

The real-time structure of MINIFØR is realized by simulating the functions of both the real-time clock and process input and output assembly language handlers with FØRTRAN subroutines. The system emulates input of process analog signals by an analog-to-digital converter (ADC) and output of analog signals to the process from digital-to-analog converters (DAC). The emulation of digital input channels allows the position of contact sensors or on-off devices to be monitored. Emulation of digital output channels allows the use of digital outputs which are either latched in the set or reset state or pulsed. User control of real-time execution is accomplished by simulating delay of execution in a program for a specified time. To control program execution and communicate with the process, the User must issue calls to system subroutines which perform the desired function. These calls conform to the standards established by the Fifth Workshop on Standardization of Industrial Computer Languages[16] and accepted by ISA[15]. Bit string manipulation per these standards is also implemented in ANSI FØRTRAN IV.

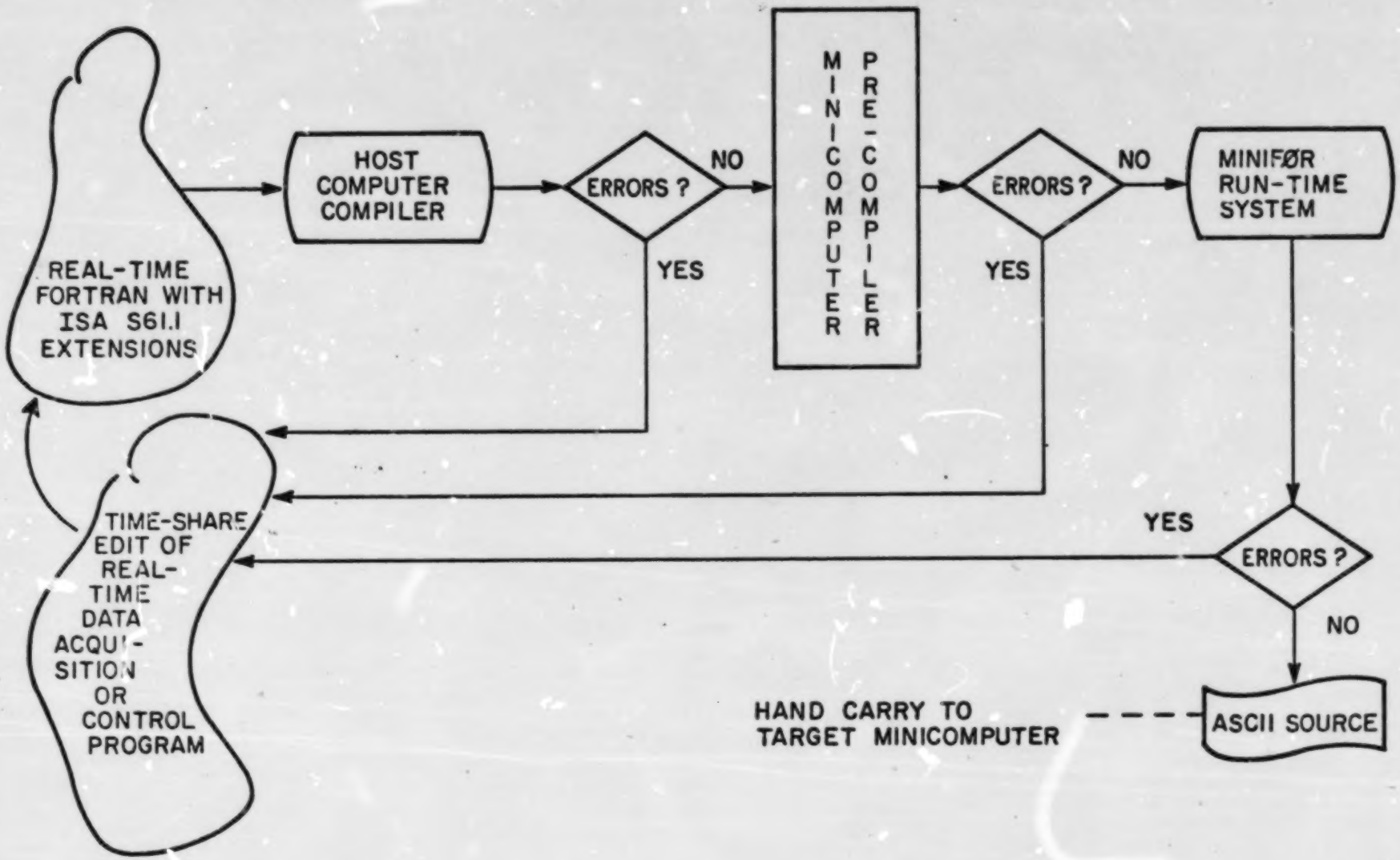

USE OF MINIFØR SYSTEM
REAL-TIME FORTRAN WITH ISA S61.1 EXTENSIONS
HOST COMPUTER COMPILER
ERRORS ?
NO
YES
MINICOMPUTER PRE-COMPILER
ERRORS?
NO
YES
MINIFØR RUN-TIME SYSTEM
TIME-SHARE EDIT OF REAL-TIME DATA ACQUI-SITION OR CONTROL PROGRAM
YES
ERRORS ?
NO
HAND CARRY TO TARGET MINICOMPUTER
ASCII SOURCE

FIGURE 2

Real-time processing is accomplished by using these standard calls to input and/or output information from/to the process. The User may, as in data acquisition systems, interrogate the process for alarm conditions or operate on process data as in chromatograph data analysis. If the application is closed-loop process control, then manipulated variables are computed and output to the process in the form of analog signals or digital outputs.

Restrictions and Limitations

The restrictions and limitations depend largely upon the type of processing performed by the User's application program and the complexity of the application and, hence, may not concern all users. Latency problems, symbol table overflow during minicomputer FØRTRAN compilation and core overflow at minicomputer load time cannot be detected. Within these limitations, MINIFØR can be utilized as a tool for debugging and developing real-time FØRTRAN IV source programs. In any event, MINIFØR is most assuredly useful as a pedagogic tool in teaching Elements of Real-Time Computing. Since the user is usually unable to recall all exceptions or extensions to ANSI FØRTRAN IV for his specific minicomputer, the precompiler should be used as illustrated in Figure 2 to ensure conformance with the minicomputer dialect.

To process a MINIFØR application, the User's application subroutine must issue calls to ISA standard input/output handlers implemented in MINIFØR. To process MINIFØR paper tape source programs on the User's minicomputer, these standard subroutines must also be available as assembly language handlers on the target mini. Currently implemented handlers are summarized in Table III.

TABLE III. MINIFØR Real-Time Handlers and Support Routines--ISA S61.1

Executive Interface	
SUBRØUTINE START (I,J,K,M)	Start Program Immediately or After Specified Time Delay
SUBRØUTINE TRNØN(I,J,M)	Start Program at Specified Time
SUBRØUTINE WAIT(J,K,M)	Delay Continuation of Program
Process I/O*	
SUBRØUTINE AISQW(I,J,K,M)	Analog Input Sequential
SUBRØUTINE AIRDW(I,J,K,M)	Analog Input Random
SUBRØUTINE DIW(I,J,K,M)	Digital Input
SUBRØUTINE AØW(I,J,K,M)	Analog Output
SUBRØUTINE DØLW(I,J,K1,K2,M)	Digital Output Latching
SUBRØUTINE DØMW(I,J,K,N,M)	Digital Output Momentary

*non-ANSI entry points also provided for AISQ,AIRD,DI,AØ,DØL, DØM

Table III continued

Bit String Manipulation	
IØR(M,N)	Inclusive OR
IAND(M,N)	Logical Product
NØT(M)	Logical Complement
IEØR(M,N)	Exclusive OR
ISHFT(M,N)	Shift Operations

See ISA S61.1[15] for details on the use of these routines

DIMENSIONed arrays in MINIFØR should be adequate for most dedicated applications, though the arrays may be increased in size if necessary through minor modifications of MINIFØR source code. Digital input and output registers are limited to 18 bits though this, too, may be easily modified.

USE OF MINIFOR IN A REAL-TIME APPLICATION

Real-time FØRTRAN program development through simulation is accomplished in four phases. First, the user must specify his exceptions and extensions to ANSI FØRTRAN IV and generate a precompiler. This procedure can be involved and will not be discussed further here. Second, the User must develop and implement a mathematical model of the physical process in the data acquisition or process control application. In most cases, simplified process models are

adequate for purposes of application program development. An indirect benefit of the whole approach proposed here is that the User is forced to analyze his data acquisition or control objectives in more detail than he might otherwise do. Third, the application program must be developed and debugged using MINIFØR and its support subroutines. Fourth, the application program is converted to ASCII punched paper tape.

Specifying the Program Input

Exclusive of input to the precompiler generator, two categories of input are required: FØRTRAN routines and input data. User supplied FØRTRAN subroutines must include a subroutine for modeling the User's real-time process and an application program for interrogating and/or controlling the process. Input data consist of specifications for describing and controlling the simulation, defining the hardware and characterizing the transducers and final control elements. In the batch mode, this must be followed by any data required by the User's application program at mini run time.

THE MØDEL Subroutine

The real-time process is simulated using a subroutine named MØDEL; the process may be described by a single subroutine or by several subroutines nested within MØDEL. This subroutine may use any feature of the host computer compiler as MØDEL is never compiled on the target mini. A User selected system

integration routine calls MØDEL many times during a simulation run. MINIFØR allows for the description of two types of multivariable nonlinear models. Algebraic models are described by;

$$\underline{Y} = \underline{F}(\underline{A},\underline{Y},T) \tag{1}$$

and dynamic models are described by:

$$\underline{YDOT} = \underline{G}(\underline{A},\underline{Y},T) \tag{2}$$

where

$\underline{Y}$ = a vector containing algebraic variables for algebraic simulations and state variables for dynamic simulations.

$\underline{A}$ = an input vector containing constants, manipulated variables, and variables to be disturbed.

$\underline{YDOT}$ = a derivative vector containing first-order derivatives of state variables.

T = independent variable time, seconds

$\underline{F}$ and $\underline{G}$ = functions for algebraic and dynamic simulations, respectively.

Any state or algebraic variable, which is to be sampled by the analog-to-digital converter, must be placed in an output vector defined by:

$$\underline{C} = \underline{H}(\underline{A},\underline{Y},T) \tag{3}$$

where C is the output vector and H is a function. The position of two-state devices, such as contact sensors or on-off devices, is described in MODEL by the vector IRELAY. A vector element with a value of one represents a closed contact or a device that is on, while a vector element with a value of zero represents an open contact or a device that is off. Positions of these devices are monitored on digital input or output channels.

FØRTRAN statements are used in MODEL to form the vectors in Equations (1) and (3). Vectors are represented by arrays of the same variable name and known functions of time are programmed utilizing the program variable T. However, the User is cautioned to avoid assigning values to T, for this is a system function. Rules governing the construction of SUBROUTINE MODEL are summarized in Table IV.

TABLE IV. Rules for Construcing Model

1. THE SUBROUTINE NAME MUST BE "MODEL"
2. MODEL MUST INCLUDE THE COMMON BLOCK: "COMMON/USER/A(30),Y(100), YDOT(100), IRELAY(128), C(25),T"
3. THE INPUT, OUTPUT, STATE AND STATE DERIVATIVE VECTORS MUST BE "A", "C", "Y" AND "YDOT" RESPECTIVELY
4. VARIABLES MUST BE ASSIGNED BY USER PRIOR TO USE
5. THE TIME VARIABLE "T" MAY BE USED BUT NOT ASSIGNED
6. THE VECTOR "IRELAY" MUST BE USED FOR DISCRETES. A"1" INDICATES ON OR CLOSED: A"O" INDICATES OFF OR OPEN
7. ADC SAMPLED VARIABLES MUST BE ASSIGNED TO VECTOR "C"

Simulation of dead time by MINIFØR is accomplished by storing values for designated Y and A variables. The User accesses "delayed" values of these variables in MODEL by using two system function subprograms, YDLY and ADLY as shown in Table V.

TABLE V. MINIFØR Dead Time Functions

Functio	Value Returned
YDLY(I,DELAY)	Y(I) at T = 0 if T $\leq$ DELAY
	Y(I) at T - DELAY if T > DELAY
ADLY(I,DELAY)	A(I) at T = 0 if T $\leq$ DELAY
	A(I) at T - DELAY if T > DELAY

Note: I is vector element; DELAY is time delay in seconds

The Application Program

The User's application may be described by one or more subroutines, but the control point for the real-time application should be APPGM with any other subroutines nested within APPGM. The User must recall that while MODEL is coded in process units, MINIFØR makes these signals available as a scaled current or voltage. Information for this conversion is supplied to MINIFØR at run time. Thus, if the User wishes to work with process units in APPGM, the simulated current or voltage signal supplied by MINIFØR must be converted to process units after input and the process units back-transformed to current or voltage prior

to output.

Controlling Execution - In practice, the real-time clock is updated by interrupt servicing. In MINIFØR, a pseudo real-time clock is advanced under control of the executive handlers in ISA S61.1. For example, WAIT delays execution in the calling program for a specified length of time by issuing a call to CLOCK, incrementing the program clock, and returning control to the requesting program. The form of this call is

```
CALL WAIT(ITIME, IUNIT, MODE)
```

where the arguments are described in Table VI.

TABLE VI. Real-Time Clock Handler[15]

```
SUBROUTINE WAIT(ITIME, IUNIT, MODE)

ITIME  SPECIFIES INCREMENTAL TIME TO WAIT BEFORE
       CONTINUING EXECUTION IN REQUESTING PROGRAM.
       FUNCTION AND UNITS ARE SPECIFIED BY IUNIT.
IUNIT  IS A TIME DELAY PARAMETER AS FOLLOWS:
       0 WAIT 'ITIME' COUNTS OF SYSTEM CLOCK
       1 WAIT FOR 'ITIME' MILLISECONDS
       2 WAIT FOR 'ITIME' SECONDS
       3 WAIT FOR 'ITIME' MINUTES
MODE   DESIGNATES AN ERROR PARAMETER
```

Process Input/Output Interfaces - When real-time processing

in a MINIFØR application involves the input or output of information to or from the process, the User must issue calls to one of the real-time input/output subroutines of Table III.

Standard input and output interfaces established by the Fifth Workshop on Standardization of Industrial Computer Languages[16] allow for two forms. The first form accommodates executive systems which permit overlapped input/output and processing; the second form accommodates executive systems which suspend execution in the requesting program until the process input or output is completed. The second form is flagged by the addition of a "W" as the last letter in the subroutine name. Only the "W" form of these handlers is simulated although MINIFØR does provide non-ANSI subroutine entry points for the overlapped I/Ø routines.

The subroutine names and argument lists conform to the process input and output standards presented in Table III. By issuing a call to one of these subroutines the User simulates transferring either analog or digital signals to or from the process. In the actual minicomputer real-time application, these same subroutines must be implemented as assembly language handlers which issue input/output transfers to the User's interfacing hardware. In MINIFØR, the functions of these assembly language handlers are simulated by FØRTRAN

subroutines having identical subprogram names and argument lists.

Analog inputs are obtained from an analog-to-digital converter. MINIFØR allows addressing up to twenty-five channels. Analog points can be input randomly using:

CALL AIRDW(NCHAN, NINCHN, NVALUE, MODE)

where these arguments are described in Table VII

TABLE VII. Analog Input Random Wait[15]

SUBROUTINE AIRDW(NCHAN, NINCHN, NVALUE, MODE)

NCHAN	SPECIFIES THE NUMBER OF ANALOG POINTS TO INPUT
NINCHN	DESIGNATES ONE-DIMENSIONAL INTEGER ARRAY FOR TERMINAL CONNECTION DATA OF THE ANALOG INPUT POINTS. SPECIFIC INFORMATION REGARDING THIS ARGUMENT IS SUPPLIED TO MINIFØR AT RUN-TIME.
NVALUE	DESIGNATES A ONE-DIMENSIONAL INTEGER ARRAY INTO WHICH THE ANALOG VALUES ARE STORED.
MODE	DESIGNATES AN ERROR PARAMETER.

Analog outputs are directed to a digital-to-analog converter. MINIFØR allows for addressing up to twenty-five channels. Analog points are output in a manner similar to that for analog input, using SUBRØUTINE AØW.

MINIFØR Input

The amount and type of data input to MINIFØR at run time is dependent upon the type of processing performed and the complexity of the simulation. Data is coded on standard data forms each of which represents a single card. Several copies of certain forms may be needed in complex simulations. A brief summary of data input requirements ire presented in Table VIII.

TABLE VIII. Batch MINIFØR - Input Data Required

- Title and Run Identification
- Simulation Specifications
- Printer Plot Requests
- Integration Rule and Parameters
- Labels for Output Variables
- Labels for Input Variables
- Hardware Specifications
- Initial Values of Input Variables
- Initial Values of State Variables
- Initial State of Discrete Registers
- System Disturbances
- Model Input Variables Delayed
- Model State Variables Delayed
- Transducer Characteristics (A(I) to ADC)
- Transducer Characteristics (C(I) to ADC)
- Final Control Element Characteristics (A(I) from DAC)
- Pulse Signals to Stepping Motors, etc.

Extensive diagnostics are used to check the input data for consistency. Default conditions exist for many items

and processing is attempted if at all possible.

MINIFØR Output

In addition to output generated by the User's application program, MINIFØR files data transferred between MØDEL and APPGM and provides for an optional dump of this information (as tables or printer plots) at the end of the simulation.

APPLICATION

Problem Statement. A simple problem in Direct Digital Control of two stirred tank chemical reactors in series illustrates the steps in a MINIFØR application. This example follows that of Coughanowr and Koppel[1] except that the analog controller is replaced by a simple DDC position algorithm. A schematic of the process is shown in Figure 3.

A feed stream, F, containing reactant R at a concentration of c_o (lb mols/ft^3) enters tank 1. Reactant R decomposes in the tanks according to an irreversible first-order reaction. The purpose of the control system is to maintain the concentration, c_2, of reactant R leaving tank 2 at a desired value despite variations of feed composition, c_o. This is accomplished by adding a stream, m, of pure R to tank 1 (F>>m). From the product stream of tank 2 a sample is continuously withdrawn to a remotely located composition analyzer. Dead time for the sample line is 30 seconds. The change in concentration for the system is represented by

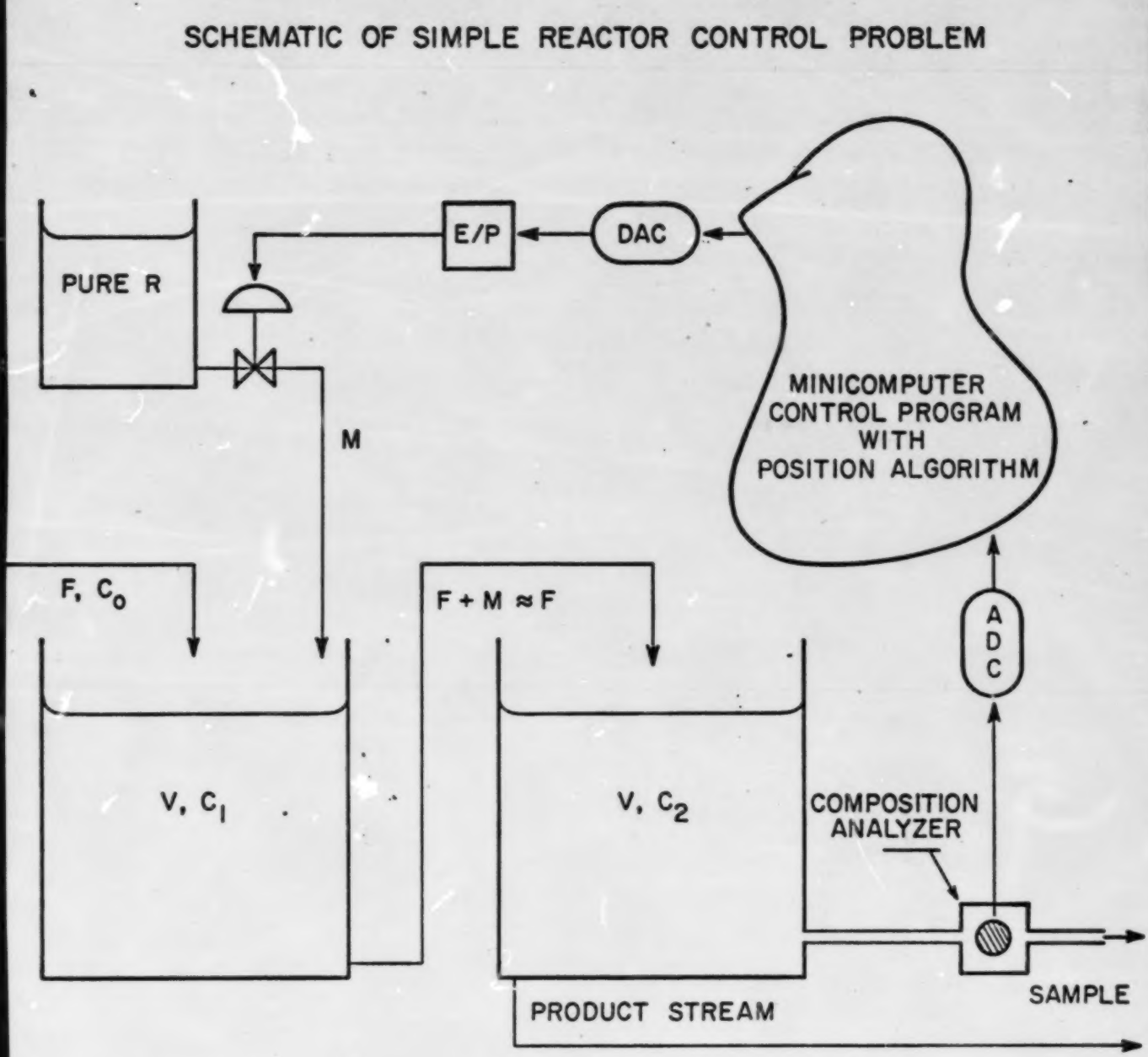

SCHEMATIC OF SIMPLE REACTOR CONTROL PROBLEM
PURE R
E/P
DAC
MINICOMPUTER
CONTROL PROGRAM
WITH
POSITION ALGORITHM
M
F, C_0
F + M ≈ F
A
D
C
V, C_1
V, C_2
COMPOSITION
ANALYZER
SAMPLE
PRODUCT STREAM

FIGURE 3.

two coupled, linear first-order differential equations:

$$\text{Tank 1:}\quad T_1 \frac{dc_1}{dt} = -c_1 + \frac{1}{(1+k_1V/F)} c_O + \frac{1}{(1+k_1V/F)} \frac{m}{F} \tag{4}$$

$$\text{Tank 2:}\quad T_2 \frac{dc_2}{dt} = -c_2 + \frac{1}{(1+k_2V/F)} c_1 \tag{5}$$

$$\text{Analysis:}\quad c_{2M} = c_2(T - 30.0) \tag{6}$$

Details of the development of these equations and the numerical values and units for all problem variables may be obtained from the original reference[1].

Model Development - Inputs to the process consist of feed composition and flow rate of pure R into tank 1. The concentration of R leaving each tank represents a state variable, while the output of the process is the concentration of R leaving the analyzer. The MØDEL for the process is constructed by making the following assignments to program variables:

$$\text{YDØT(1)} \hat{=} \frac{dc_1}{dt};\quad \text{YDØT(2)} \hat{=} \frac{dc_2}{dt};\quad \text{Y(1)} \hat{=} c_1;\quad \text{Y(2)} \hat{=} c_2 \tag{7}$$

$$\text{YDLY(2, 30.0)} \hat{=} c_{2M};\quad \text{A(1)} \hat{=} c_O;\quad \text{A(2)} = m \tag{8}$$

MINIFØR has the option to store all elements of the A and C vectors at user specified internvals during the simulation. These values are printed in tabular or printer-plot form at the end of the simulation as a diagnostic aid. Since values of the state vectors Y and YDOT are not stored, values of interest must be assigned to positions in the C vector. In all cases, if an analog variable is to be sampled by the ADC, it must be assigned a position in the C vector. Assume that concentrations in both tank effluents as well as analyzer output are of interest. The following FØRTRAN replacement statements must be used in MØDEL:

```
C(1) = Y(1); C(2) = Y(2);  C(3) = YDLY(2, 30.0)          (9)
```

The complete FØRTRAN IV coding for subroutine MØDEL is shown in Table IX. All variables are in process units. MINIFØR converts these signals to current or voltage as determined by the User hardware definitions.

TABLE IX. MØDEL FØR CØNTRØL ØF STIRRED TANK REACTØR

```
      SUBRØUTINE MØDEL
      CØMMØN/USER/A(30), Y(100), YDØT(100)
      CØMMØN/USER/IRELAY(128), C(25),T
C     CØNCENTRATIØN GRADIENT, TANK 1
      YDØT(1) = (-Y(1) + 0.666*A(1))/120.0
     1  + (0.401*A(2)*0.00178)/120.0
C     CØNCENTRATIØN GRADIENT, TANK 2
      YDØT(2) = (-Y(2) + 0.33288*Y(1))/60.0
C     ASSIGN ØUTPUT VARIABLES TØ ADC
      C(1) = Y(1)
      C(2) = Y(2)
C     DELAY TANK 2 CØNCENTRATIØN
      C(3) = YDLY(2,30.0)
      RETURN
      END
```

Table X illustrates (in the form output by MINIFØR) definition of an analog signal "sampled" on Channel 51. The MØDEL variable, C(3), corresponds to the process analyzer hardware pin connection on Channel 51. The units and range of the measurand are given. The table entry is the signal distribution (here, 10 to 50 ma DC on a linear scale) transmitted to the ADC. Table XI illustrates definition of an analog signal "output" on Channel 110. The MØDEL variable A(2), corresponds to the process final control element hardware pin connection on Channel 110. Here, the table entry is in gpm corresponding to a given percent above the base signal of 10 ma DC to the final control element. Note the non-linear control valve characteristic.

APPGM Development - The user's application may be described by one or more routines, but the control point for the real-time application should be APPGM with any subroutines nested within APPGM. The user must recall that while MØDEL is coded in process units, MINIFØR makes these signals available as a scaled current or voltage. Information for this conversion is supplied to MINIFØR at run time. Thus, if the User wishes to work with process units in APPGM, the simulated current or voltage signal supplied by MINIFØR must be converted to process units after input and the process units back-transformed to current or voltage prior to output.

The process control application for the example of Figure

TABLE X

ADC SPECIFICATION FOR MEASURED CONCENTRATION

SPECIFIED OUTPUT VARIABLES TO BE MEASURED BY ANALOG TO DIGITAL CONVERTER

PROCESS SPECIFICATIONS				ANALOG SIGNAL TO ADC FOR DESIGNATED LEVEL OF MEASURAND, HDWE UNITS					
VARIABLE MEASURED	UNITS	MINIMUM MEASURAND	MAXIMUM MEASURAND	0 PERCENT	30 PERCENT	50 PERCENT	70 PERCENT	100 PERCENT	SAMPLED ON CHANNEL
C (3)	CONC	.01	.05	10.00	22.00	30.00	38.00	50.00	51

TABLE XI

DAC SPECIFICATION FOR MANIPULATED FLOWRATE

SPECIFIED INPUT VARIABLES TO BE MANIPULATED BY DIGITAL TO ANALOG CONVERTER

PROCESS SPECIFICATIONS							HARDWARE SPECIFICATIONS		
		VALUE OF MANIPULATED VARIABLE FOR DESIGNATED LEVEL OF DAC OUTPUT							
INPUT VARIABLE MANIPULATED	UNITS	0 PERCENT	30 PERCENT	50 PERCENT	70 PERCENT	100 PERCENT	MINIMUM SIGNAL TO FCE	MAXIMUM SIGNAL TO FCE	SIGNAL OUTPUT ON CHANNEL
A (2)	GPM	0.00	2.20	4.50	8.80	20.00	10.00	50.00	110

3 consists of a single DDC loop with proportional plus integral control action. The complete code for APPGM is shown in Table XII.

TABLE XII

CONTROL PROGRAM FOR STIRRED TANK REACTOR

```
      DIMENSIØN NINCHN (1), NØUTCH (1), NVALUE (1)
      DATA XKC, TI, XMAN/500.0, 10.0, 38.672/
      DATA SETPT, SUM/0.0244, 0.0/
C     INITIALIZE HARDWARE CØNNECTIØNS
      NINCHN(1) = 51
      NØUTCH (1) = 110
      GØ TØ 10
C     WAIT 5 SECØNDS TØ SAMPLE
    5 CALL WAIT (5,2, MØDE)
C     GET CØNCENTRATIØN ØF TANK 2 EFFLUENT
   10 CALL AIRDW (1, NINCHN, NVALUE, MØDE)
C     CØNVERT ADC MA*1000 TØ CØNCENTRATIØN UNITS
      CVAR = FLØAT (NVALUE(1))/1.0E+06
C     GET NEW VALVE PØSITIØN IN MA*1000
      ERRØR = SETPT - CVAR
      SUM = SUM + ERRØR
      VALPØS = 1000.0* (XMAN + XKC* (ERRØR + SUM/TI))
C     ØUTPUT NEW VALVE PØSITIØN
      NVALUE (1) = VALPØS
      CALL AØW (1, NØUTCH, NVALUE, MØDE)
C     GET NEXT SAMPLE
      GØ TØ 5
      END
```

37

Note that NINCHN, NOUTCH and NVALUE in APPGM are DIMENSIONed even though there is only one element needed in each. This is necessary in this case because the target mini routines AIRDW and AØW expect these arguments to be DIMENSIONed and function improperly if they are not. The DATA statements initialize controller gain (XKC), integral time (TI) and manual reset (XMAN) as well as the set point (SETPT). Pin connections are identified for the single ADC input Channel 51 and the single DAC output Channel 110. One value of the analyzer output is obtained by a CALL AIRDW and converted to process units. A simple proportional plus integral control algorithm calculates the next value position to be output by a CALL AØW. Execution is suspended for 5 seconds whereupon the cycle repeats. The real-time clock is initialized by the mini I/Ø operating system on the first call to WAIT.

Results. Figure 4 shows system behavior in process units. Note that c_o (lower left hand corner of Figure 4) starts at 0.10, decreases to 0.05 at 60 sec. and increases back to 0.10 at 240.sec. Thus a short closed square pulse is generated.

Note the response to this disturbance. As expected, the concentration measured by the analyzer shows no change for 30 seconds due to the pure transport lag, then drops, begins to recover under control, overshoots the set point and

MINIFØR OUTPUT FOR REACTOR CONTROL PROBLEM

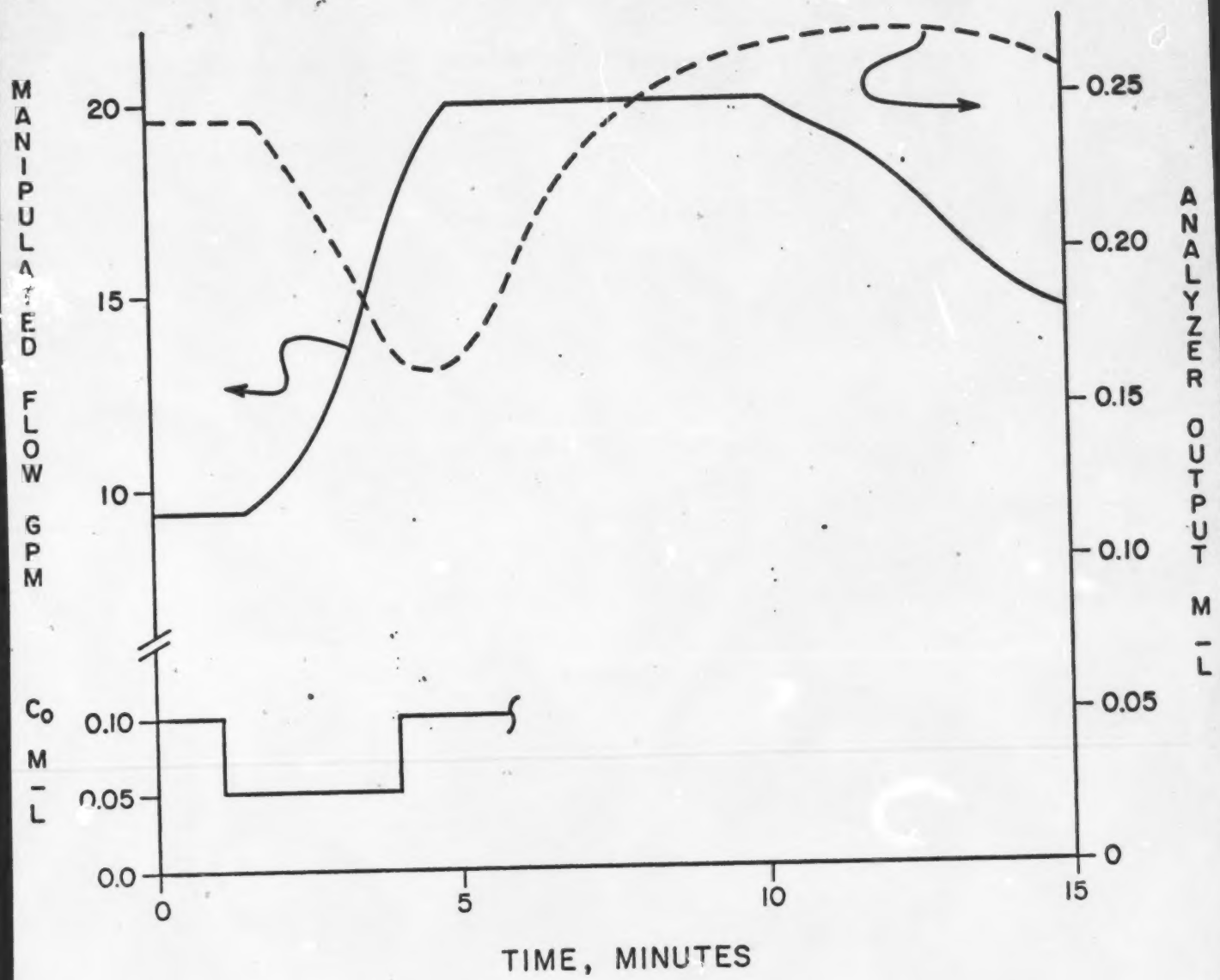


FIGURE 4.

finally settles in with the long tail characteristic of a second order response.

Note also that the manipulated variable saturates at 20 gpm as this was the maximum permissible value specified to MINIFØR. Thus the simulated interface, transducers and final control elements prevent development of "good" control programs based on physically unrealizable values of process variables.

This example illustrates a very simple and straight forward application. Much more complicated cases have been handled with little difficulty[6].

CONCLUSION

In spite of its limitations, MINIFØR can aid program development for minimum configuration dedicated minicomputer data acquisition and process control applications. Minicomputer compilation errors are trapped by a host computer driven general purpose precompiler. System real-time structure is built around standard executive and process input/output interfaces established by the Fifth Workshop on Standardization of Industrial Computer Languages and accepted by ISA. Paper tape FØRTRAN IV source programs for processing on the actual minicomputer application are produced by a card-to-tape conversion program. MINIFØR can be a productive tool for debugging, developing, and updating real-time FØRTRAN

programs used in industrial applications. Further, the system has potential for wide usage in an academic environment for unit operation exercises and studies in process control. Industrial acceptance of such methodology depends upon vendor implementation of the standard interface calls published as ISA standards. Users must recognize the advantages of standardized process calls and demand such vendor implementation, at least as a subset of the options available.

ACKNOWLEDGEMENT

Thanks are due the Arizona Research Foundation; the University of Arizona Computing Center; and the National Science Foundation under Grant EPP-74-21344 for their support of various phases of this work.

LITERATURE CITED

1. Coughanowr, D. R., and Koppel, I. D., 1965. Process Systems Analysis and Control, pp. 123-129, McGraw Hill Co., New York City, New York.

2. Jarvis, P. H., "Some Experiences with Process Control Languages", IEEE Trans. on Indust. Electorn. and Control Instrum. 15:54-56, 1968.

3. Kaenel, Reg. A., "Minicomputers--A Profile of Tomorrow's Component", IFEE Trans. Audio Electroacoust. 18:354-379, 1970.

4. Mensh, M., and Diehl, W., "Extended FØRTRAN for Process Control", IEEE Trans. on Indust. Electron. and Control Instrum. 15:75-79, 1968.

5. Murphy, John A., "Minicomputers", Modern Data 5:58-72, 1971.

6. Peterson, T. W., J. W. White and E. E. Krist, "Off-Line Development and Testing of a Minicomputer Process Control Program for an Industrial Grinding Circuit", AIME Preprint 74-B-40, Feb., 1974.

7. Roberts, Bert C., "FØRTRAN IV in a Process Control Environment", IEEE Trans. on Indust. Electorn. and Control Instrum. 15:61-63, 1968.

8. Schneck, P. B., and E. Angel, "A FØRTRAN to FØRTRAN Optimizing Compiler", Computer J., 16(4): 322-330, 1972.

9. Spencer, Henry W., Shepardson, Harlon P., and McGowan, Larry M., "Small Computer Software", IEEE Comput. Group News 3:15-20, 1970.

10. White, J. W., G. D. Ripley and W. L. Hutchison, "A FØRTRAN Based Scanner and SLR(1) Parser-Generator for Minicomputer Precompilers". In Preparation.

11. Williams, T. J. (ed.). Emulator, Simulator, and Translator Programs for the Small Digital Computers Necessary for Medical Laboratory Automation, Purdue Laboratory Lafayette, Indiana, 1970.

12. Williams, T. J., Contribution to: "Interim Report of the CACHE Real-Time Task Force, p.16, Oct., 1973.

13. Advertisement, Digital Equipment Corporation. Electron. 45:29, 1972.

14. "Executive Summary of MODERN DATA'S 1973 Minicomputer Market Survey", Modern Data Services, Inc., Framingham, Massachusetts, 1973.

15. Industrial Computer System FØRTRAN Procedures for Executive Functions and Process Input-Output, ISA Standard S61.1, Instrument Society of America, Nov., 1972.

16. Minutes, Fifth Workshop on Standardization of Industrial Computer Languages. Purdue Laboratory for Applied Industrial Control, Purdue University, Lafayette, Indiana. May, 1971.

17. Minutes, Sixth Workshop on Standardization of Industrial Computer Languages. Purdue Laboratory for Applied Industrial Control, Purdue University, Lafayette, Indiana. Oct., 1971.

18. Minutes, Seventh Workshop on Standardization of Industrial Computer Languages. Purdue Laboratory for Applied Industrial Control, Purdue University, Lafayette, Indiana, April, 1972.

19. Minutes, Eighth Workshop on Standardization of Industrial Computer Languages. Purdue Laboratory for Applied Industrial Control, Purdue University, Lafayette, Indiana, October, 1972.

20. Minutes, Ninth Workshop on Standardization of Industrial Computer Languages. Purdue Laboratory for Applied Industrial Control, Purdue University, Lafayette, Indiana, May, 1973.